Monetizing Machine Learning

Quickly Turn Python ML Ideas into Web Applications on the Serverless Cloud

Manuel Amunategui
Mehdi Roopaei

Apress®

Monetizing Machine Learning: Quickly Turn Python ML Ideas into Web Applications on the Serverless Cloud

Manuel Amunategui
Portland, Oregon, USA

Mehdi Roopaei
Platteville, Wisconsin, USA

ISBN-13 (pbk): 978-1-4842-3872-1
ISBN-13 (electronic): 978-1-4842-3873-8
https://doi.org/10.1007/978-1-4842-3873-8

Library of Congress Control Number: 2018956745

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Susan McDermott
Development Editor: Laura Berendson
Coordinating Editor: Rita Fernando

Cover designed by eStudioCalamar

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484238721. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Authors

Manuel Amunategui is VP of Data Science at SpringML, a Google Cloud and Salesforce preferred partner, and holds Masters in Predictive Analytics and International Administration. Over the past 20 years, he has implemented hundreds of end-to-end customer solutions in the tech industry. The experience from consulting in machine learning, healthcare modeling, six years on Wall Street in the financial industry, and four years at Microsoft, has opened his eyes to the lack of applied data science educational and training material available. To help alleviate this gap, he has been advocating for applied data science through blogs, vlogs, and educational material. He has grown and curated various highly focused and niche social media channels including a YouTube channel (`www.youtube.com/user/mamunate/videos`) and a popular applied data science blog: amunategui.github.io (`http://amunategui.github.io`).

Mehdi Roopaei (M'02–SM'12) is a Senior Member of IEEE, AIAA, and ISA. He received a Ph.D. degree in Computer Engineering from Shiraz University on Intelligent Control of Dynamic Systems in 2011. He was a Postdoctoral Fellow at the University of Texas at San Antonio, 2012-Summer 2018, and holds the title of Assistant Professor at the University of Wisconsin-Platteville, Fall 2018. His research interests include AI-Driven Control Systems, Data-Driven Decision Making, Machine Learning and Internet of Things (IoT), and Immersive Analytics. He is Associate Editor of IEEE Access and sits on the Editorial Board of the IoT Elsevier journal. He was guest editor for the special issue: "IoT Analytics for Data Streams" at IoT Elsevier and published a book *Applied Cloud Deep Semantic Recognition: Advanced Anomaly Detection* (CRC Press, 2018). He was IEEE chapter chair officer for joint communication and signal processing communities at San Antonio, Jan-July 2018. He has more than 60 peer-reviewed technical publications, serves on the program committee at several conferences, and is a technical reviewer in many journals.

About the Technical Reviewers

Rafal Buch is a technologist living and working in New York as a financial systems architect. He's been doing software engineering for two decades and spends most of his free time hacking in coffee shops and exploring new technologies. Blog: rafalbuch.com

Matt Katz has been working in financial technology since 2001 and still gets excited about new stuff all the time. He lives online at `www.morelightmorelight.com` and he lives offline in New York with his two strange children and one amazing, patient wife.

Acknowledgments

To the friends, family, editors, and all those involved in one way or another in helping make this project a reality–a huge thanks! Without your help, this book would have never seen the light of day.

Introduction

A few decades ago, as a kid learning to program, I had an ASCII gaming book for my Apple II (of which the name eludes me) that started each chapter with a picture of the finished game. This was the teaser and the motivator in a book that was otherwise made up of pages and pages of nothing else but computer code. This was years before GitHub and the Internet. As if it were only yesterday, I remember the excitement of racing through the code, copying it line-by-line, fixing typos and wiping tears just to play the game. Today, a lot has changed, but even though the code is downloadable, we put a screenshot of the final product at the beginning of each chapter, so you too can feel the motivation and excitement of working through the concepts.

Low-Barrier-To-Entry and Fast-To-Market

This book will guide you through a variety of projects that explore different Python machine learning ideas and different ways of transforming them into web applications. Each chapter ends with a serverless web application accessible by anyone around the world with an Internet connection. These projects are based on classic and popular Python data science problems that increase in difficulty as you progress. A modeling solution is studied, designed, and an interesting aspect of the approach is finally extended into an interactive and inviting web application.

Being a data scientist is a wonderful profession, but there is a troubling gap in the teaching material when trying to become one. Data science isn't about statistics and modeling; it is about fulfilling human needs and solving real problems. Not enough material tackles the big picture. it seems that whenever you start talking about the big picture, you have to sign a non-disclosure agreement (NDA). This is an essential area of study and if you are like me, you need to understand why you are doing something in order to do it right. These aren't difficult topics, especially when you use the right tools to tackle them.

We won't focus on "**becoming a data scientist**" as an end goal; there are plenty of books on that topic already. Instead, we'll focus on getting machine learning products to market quickly, simply, and with the user/customer in mind at all times! That's what is missing in this profession's educational syllabus. If you build first and then talk to your customer, your pipelines will be flawed and your solutions will miss their target. I have redrawn Drew Conway's Data Science Venn Diagram with the customer as top priority (Figure 1).

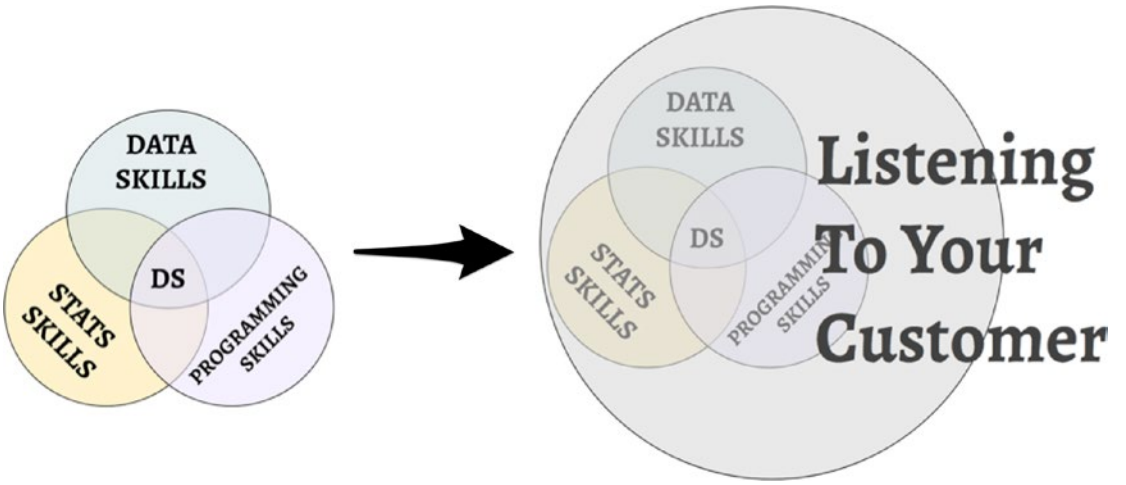

Figure 1. *The classic data science Venn diagram next to my updated version*

[1]Mehdi and I worked hard on the content of this book. We took our time to develop the concepts, making sure they were of practical use to our reader (i.e., our customer–always keep the customer in mind at all times). I built the material and Mehdi edited it. This is an ambitious book in terms of scope and technologies covered. Choices and compromises had to be made to focus on the quickest ways of getting practical use out of the material. The tools are constantly changing. Some things in this book are going to be stale by the time you read them, and that is OK (you can go to the GitHub repo for updates). Here, everything changes all the time, but things tend to change for the better! So, learning new tricks often means learning better, faster, and more powerful ways to do things. This book will not only show you how to build web applications but also point you in the right direction to deepen your knowledge in those areas of particular interest.

[1]http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

If this was a class, I'd have you sign a "**compete agreement**": yes, the opposite of a non-compete. I would have you go through this book, understand the tools, and then copy them and make them your own. These are meant to be templates to quickly get your platforms up and running to focus on the bigger things, to build impactful tools for your customers and friends. When you understand this, that's the day you graduate with all the entitlements and privileges of being called a "**data science professional.**"

What is the Serverless Cloud?

Cloud providers have gone to great efforts to improve web hosting solutions and bring costs down. The recent advent of the "**serverless**" option, which abstracts a large swath of the configuring process, is available on three of the four cloud providers covered in this book. This means you can get your projects up and running on fully managed platforms, with automatic load-balancing, throughput scaling, fast deployments, etc., without having to select, configure, or worry about any of it. The level of disengagement with these architectural and monitoring options is up to you. You can choose what you want to control and what you want to delegate to the provider. One thing is guaranteed: the site will automatically adjust with traffic and offer unparalleled uptime.

This allows us to focus on what is important without getting bogged down by the trappings and support needed to get there. These so-called "**trappings**" are critical and need to be taken very seriously. This is why we are looking at four reputable cloud providers that will give us the peace of mind required to fully focus on our web applications and not worry about the site crashing or the provider going dark. Let's focus on what is important and get to work!

Critical Path in Web Application Development

So many machine learning models stagnate in their original coded state: hard to understand, with no easy way to invite others to benefit from its insights. These models are doomed to be forgotten. Even those that manage to escape the confines of an integrated development interface fall short of their potential when reduced to a static

chart or a cryptic modeling score. This book aims to avoid this trap by breaking down the process of extending a machine learning model into a universally accessible web application. Each chapter follows these three critical steps:

1. **Modeling the right way**. We start at the end, by understanding what users want to see, and by investing time and thought on the final goal and user experience. We ensure that the appropriate modeling approach is used in order to reach a web-application state rapidly and without surprise (Figure 2).

Figure 2. *Always check that there is an audience for your idea before building (source Lucas Amunategui)*

2. **Designing and developing a local web application**. This step requires leveraging various web front-end technologies to offer the needed interactivity and dynamism to highlight a model's insight and sustain a user's interest. The final product at this stage is indistinguishable from the next one except that it is hosted on your local machine, not the cloud.

3. **Deploying onto a popular and reliable serverless cloud provider**. Each provider has unique requirements, advantages, and disadvantages that need to be understood and addressed. This is the final stage where the world gets to enjoy and learn from your work.

We start by tackling easy ways of offering intelligent interactivity, like leveraging a model's coefficients or saving a trained model, then move to the complex, like using a database to track engagement or relying on open-source pretrained models for image recognition. A fictional case study around stock market predictions is started in the first section, then revisited in subsequent ones. New features are added to it until it culminates into a complex dashboard with a paywall to offer customized intelligence to paying subscribers.

By focusing on classic, data science problems, coupled with popular, open-source technologies like Python, Flask, Ajax, Bootstrap, JavaScript, and HTML, you should find yourself on familiar ground and if you don't, you'll have a drastically reduced learning curve. We focus on simple tools and simple techniques to reliably and rapidly get machine learning ideas out into the wild. The tools and approaches are revisited in each chapter, so don't worry if some aspects aren't clear from the start; keep going and things will keep getting clearer.

We also rotate cloud providers in each chapter, so you will get exposed to the most popular providers. This will give you plenty of insights into which provider to select for your future project. I recommend going through all chapters, as each will show different ways of doing things, highlighting a provider's strengths along with showing unique tips and tricks to get things done quickly.

You, the Reader

This book is for those interested in extending statistical models, machine learning pipelines, data-driven projects, or any stand-alone Python script into interactive dashboards accessible by anyone with a web browser. The Internet is the most powerful medium with an extremely low barrier to entry–anybody can access it, and this book is geared to those who want to leverage that.

This book assumes you have Python and programming experience. You should have a code editor and interpreter in working order at your disposal. You should have the ability to test and troubleshoot issues, install Python libraries, and be familiar with popular packages such as NumPy, Pandas, and Scikit-learn. An introduction to these basic concepts isn't covered in this book. The code presented here uses Python 3.x only and hasn’t been tested for lower versions. Also, a basic knowledge of web-scripting languages will come in handy.

This book is geared towards those with an entrepreneurial bent who want to get their ideas onto the Web without breaking the bank, small companies without an IT staff, students wanting exposure and real-world training, and for any data science professional ready to take things to the next level.

How to Use This Book

Each chapter starts with a picture of the final web application along with a description of what it will do. This approach serves multiple purposes:

- It works as a motivator to entice you to put in the work.
- It visually explains what the project is going to be about.
- And more importantly, it teaches how critical it is to have a clear customer-centric understanding of the end game whenever tackling a project.

The book will only show highlights of the source code, but complete versions are attached in the corresponding repositories. This includes a Jupyter notebook when covering data exploration and zipped folders for web applications.

The practical projects presented here are simple, clear, and can be used as templates to jump-start many other types of applications. Whether you want to understand how to create a web application around numerical or categorical predictions, the analysis of text, the creation of powerful and interactive presentations, to offer access to restricted data, or to leverage web plugins to accept subscription payments and donations, this book will help you get your projects into the hands of the world quickly.

Note For edits/bugs, please report back at www.apress.com/9781484238721.

Tools of the Trade and Miscellaneous Tips

Here is a brief look at the tools that will transform our machine learning ideas into web applications quickly, simply, and beautifully. This isn't meant to be a comprehensive or complete list, just a taste of the different technologies used and pointers for you to follow if you want to learn more (and I hope you will).

Jupyter Notebooks

The book only shows code snippets, so you need to download and run the Jupyter notebook for each chapter in order to run things for yourself and experiment with the various features and different parameters. If you aren't familiar with Jupyter notebooks, they are web-based interactive Python interpreters great for building, tweaking, and publishing anything that makes use of Python scripting. It attaches to a fully functioning Python kernel (make it a Python 3.x one) and can load and run libraries and scripts just like any other interpreter. To install Jupyter notebooks, follow the official docs at http://jupyter.readthedocs.io/en/latest/install.html.

There are various ways to install it, including the "**pip3**" command; check official documentation for the different ways of doing it if this approach doesn't work for you (Listing 1).

Listing 1. Install Jupyter

```
sudo pip3 install jupyter
```

To use a Jupyter notebook is both easy and powerful at the same time. You simply need to download the notebook to your local machine (it will be the file with a *.ipynb extension), open a command/terminal shell window, navigate to that folder, and run the "**notebook**" command (Listing 2).

Listing 2. Run a Notebook (check official docs for alternative ways of starting notebooks)

```
jupyter notebook
```

This command will open a web page showing the content of the folder from where it was launched (Figure 3). You can navigate down a folder structure by clicking the folder icon right above the file listings.

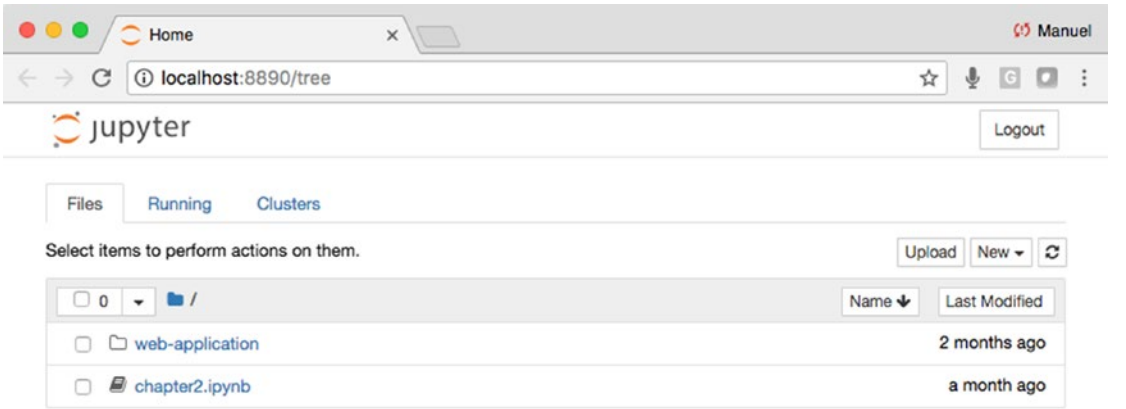

Figure 3. *Jupyter notebook landing page*

To open a Jupyter notebook, simply click any file with the "*.**ipynb**" extension and you are good to go! If you want to create a brand-new notebook, click the "**new**" button at the right of the dashboard next to the refresh button.

Note For additional information, issues with Jupyter notebooks, and attaching kernels, see: http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html.

Flask

Flask is a lightweight but very powerful server-side web framework. It is the brains behind all the applications presented in this book. It is also the glue between our Python data producing functions and web pages. That is one of the reasons I love working with Flask, as it allows us to link stand-alone Python scripts to server-side web frameworks without leaving the Python language; it makes passing data between objects a whole lot easier!

Flask comes with the bare minimum to get a web page published. If you need additional support, like a database, form controls, etc., you will have to install additional libraries and that is why it is called a lightweight, microframework. This is also what makes it so easy to use, as you only have to learn a few more tricks; everything else uses the tried-and-true Python libraries that we're already familiar with.

Unfortunately, we can only work in Python for so long and eventually you will need to step into front-end web scripting. But don't let that bother you; there are so many great examples on the Web (Stackoverflow.com, w3schools.com) and the incredible looking GetBootstrap.com templates to get you there as quickly as possible.

Note For more information on Flask, check the official Flask documentation at `http://flask.pocoo.org/`.

HTML

HTML, which means Hypertext Markup Language, needs no introduction. This is the lowest common denominator in terms of web technologies. It has been around for years and is used to create practically all web pages and web applications in existence.

For those wanting to brush up on this topic, the amount of free material on the Web can be overwhelming. I recommend anything from w3schools.com. Their material is well organized, comprehensive, and often interactive.

CSS

Cascading Style Sheets (CSS) is what makes most websites out there look great! We use two types of CSS files here: the CSS links loaded in the "<**HEAD**>" section of most web pages (the most common) and custom CSS as shown in code snippet in Listing 3.

Listing 3. Custom CSS Script Block

```
<STYLE>
.btn-circle.btn-xl {
    width: 70px;
    height: 70px;
    padding: 10px 2px;
    border-radius: 35px;
    font-size: 17px;
    line-height: 1.33;
}
</STYLE>
```

The CSS files that are hosted on outside servers cannot be customized but are usually best-in-class. But there are times you simply need to customize a feature on your page, and that is when you create a local CSS file or a style tag directly in the HTML page. It is then applied to a particular tag or area using the "**class**" parameter (Listing 4).

Listing 4. Applying CSS Tag to an HTML Tag

```
<button type="button" onclick="calculateBikeDemand(this)"
id="season_spring" class="btn btn-info btn-circle btn-xl">
<i class="fa fa-check">Spring</i></button>
```

CSS defines in great detail what size, color, font, everything and anything under the sun, should look like. It also allows the generalization of your look-and-feel through your web portal. You create it once and can have all your pages invoke it to inherit that particular style.

Note For additional information and training on CSS, check out the w3schools.com.

Jinja2

Jinja2 is used to generate markup and HTML code, and works tightly with Flask variables. It is created by Armin Ronacher, and is widely used to handle Flask-generated data and if/then logic directly in HTML templates.

In this HTML template example, a Flask-generated value called "**previous_slider_value**" is injected into the slider's "**value**" parameter using Jinja2. Note the use of double curly brackets (Listing 5).

Listing 5. Jinja2 Passing Data to HTML Input Control

```
<input type="range" min="1" max="100" value="{{previous_slider_value}}"
id="my_slider">
```

Note For additional information on Jinja2, check out the docs at http://jinja.pocoo.org/docs/2.10/.

JavaScript

JavaScript is a real programming language in and of itself. It can add extremely powerful behavior to any of your front-end controls. JavaScript brings a great level of interactivity to a web page, and we will leverage it throughout each chapter.

Here is an interesting example where we capture the mouse-up event of an HTML slider control to submit the form to the Flask server. The idea is that whenever a user changes the slider value, Flask needs to do some server-side processing using the new slider value and regenerate the web page (Listing 6).

Listing 6. JavaScript Capturing Slider "**onmouseup**" Event

```
slider1.onmouseup = function ()
{
    document.getElementById("submit_params").submit();
}
```

Note For additional information and training on JavaScript, check out w3schools.com.

---

jQuery

JQeury is a customized JavaScript library to facilitate the handling of complex front-end and behavior events, and insures compatibility between different browser versions.

jQuery will facilitate things like button, drop-down dynamic behavior, even Ajax (a critical technology used heavily in many of this book's projects).

Note For more information on JQuery, check out the official documents at JQuery.com.

Ajax

Ajax is a great front-end scripting technique that can add dynamic server-side behavior to a web page. It allows sending and receiving data without rebuilding or reloading the entire page as you would do with form submits. One area where it is commonly used is on map web pages, such as Google Maps, which allows dragging and sliding the map without reloading the entire page after every move.

Note For additional information and training on Ajax, check out w3schools.com.

Bootstrap

Bootstrap is a very powerful, almost magical tool for front-end web work. It is used by almost 13% of the Web according to BuiltWith Trends.[2] It contains all sorts of great looking styles and behavior for most web tags and controls. By simply linking your web page to the latest Bootstrap, CSS will give any boring HTML page an instant and professional looking makeover!

If you look at any of the HTML files in this book, the first thing you will notice are the links wrapped in "**LINK**" and "**SCRIPT**" tags at the top of the page. These represent an enormous shortcut in building a web page (Listing 7).

[2] https://trends.builtwith.com/docinfo/Twitter-Bootstrap

Listing 7. Link Tag to Inherit Bootstrap CSS Styles

```
<LINK rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/4.0.0/css/bootstrap.min.css">
```

All the HTML files in this book (and more than likely any web page you will create in the future) will use these links to download premade bootstrap and JavaScript scripts and automatically inherit the beautiful fonts, colors, styles, and behaviors that are prevalent all over the Internet. This allows you to immediately inherit best-in-class looks and behavior with minimal effort.

Note For additional information and training on Bootstrap, check out the official documents on GetBootstrap.com.

Web Plugins

Web plugins have a huge advantage: they push a large swath of hardware, data, and/or security management onto someone else, preferably someone specialized in that area. There is no reason to reinvent the wheel, waste valuable time, or introduce security risks. Let others take care of that and focus on what you do best; this is what this book is all about!

Unfortunately, we only have the bandwidth to explore a few of these, but here is a list of good ones that I've either used in the past or heard good things from others (and there are hundreds more out there that are probably as good–look for those that offer good terms for small businesses along with demo or test accounts to experiment before committing).

Membership Platforms

There are several platforms to note.

Memberful (`www.memberful.com`)

Memberful is the plugin we will work with and implement in this book. I personally really like Memberful.com and think it is a great choice for anybody looking for an easy way to manage a paywall section of a website. It offers credit card payment through Stripe.com, offers user-management features, and is tightly integrated within your own web application.

Patreon (www.patreon.com)

Patreon is a membership platform and plugin for artists and content creators.

Wild Apricot (www.wildapricot.com)

Wild Apricot is a membership platform for small and nonprofit organizations.

Subhub (www.subhub.com)

Subhub is a membership platform designed for entrepreneurs, experts, and organizations.

Membergate (www.membergate.com)

Membergate is a platform for corporate communications, newsletters, associations, and restricted access sites.

Payment Platforms

There are several platforms available.

Paypal Donations (www.paypal.com/us/webapps/mpp/donation)

I've used Paypal plugins in the past and have been delighted with the ease of installation and use. All you need is a Paypal account in good standing and the rest is a cinch.

Paypal Express (www.paypal.com/us/webapps/mpp/express-checkout)

Paypal Express is still Paypal but for quick and easy checkouts.

Stripe (http://stripe.com/)

Stripe is a payment options that easily allow websites to accept online credit card payments. It is the payment engine behind Memberful.com that we will see in the last chapter of this book.

Analytics

Building your own web-usage tracker requires a lot of Flask custom code on every page, along with a database to save those interactions and an analytical engine to make sense of it. That's a lot of work! Instead, with Google Analytics, all we have to do is add a JavaScript snippet of code at the top of each page. It is free for basic analysis, which is fine for our needs.

Message Boards

I have used https://disqus.com in the past to add message boards to static websites. It creates the appearance of professional-looking message boards directly on your site, all the while being managed elsewhere.

Mailing Lists

I have used formspree.io for many years and love it! It is trivial to add to any static web page along with a text box and submit button. Users can add their email address on your web page and https://formspress.io will email you the submitted information. This is a great option if you are hosting a static site or don't want to deal with managing your own database.

Git

Git is a great version control tool; it allows you to store your code's creation, changes, updates, and any deletions happening in a repository. It is tightly integrated with GitHub, which is critical for code safeguard and collaboration. It is also integrated on most of the cloud providers out there. In some chapters we will use it and in others we won't. If you end up working on larger applications or collaborate with others, I highly recommend you start using it.

Most cloud providers support online code repositories like GitHub, BitBucket, and others. These online repos work with Git, so learning the basics will give you a big leg up. The process of deploying web applications on Microsoft Azure is tightly integrated with Git, so please take a look at this basic primer or go online for some great tutorials such as try.github.io[3]:

- **git init**: creates a local repository
- **git clone https://github.com/**... clones a GitHub repository to your local drive
- **git status**: list files that are changed and awaiting commit + push to repo
- **git add .**: add all files (note period)
- **git add '*.txt'**: add all text files
- **git commit**: commit waiting files
- **git log**: see commit history

[3] https://try.github.io/

- **git push (or git push azure master)**: push branches to remote master
- **git pull**: get remote changes to local repo
- **git reset ***: to undo git
- **git rm --cached <file>**: stop tracking a file

Virtual Environments

Using a virtual environment offers many advantages:

- Creates an environment with no installed Python libraries
- Knows exactly which Python libraries are required for your application to run
- Keeps the rest of your computer system safe from any Python libraries you install in this environment
- Encourages experimentation

To start a virtual environment, use the "**venv**" command. If it isn't installed on your computer, it is recommended you do so (it is available via common installers such as pip, conda, brew, etc). For more information on installing virtual environments for your OS, see the "**venv - Creation of virtual environments**" user guide: https://docs.python.org/3/library/venv.html.

Open a command window and call the Python 3 "**venv**" function on the command line to create a sandbox area (Listings 8 and 9).

Listing 8. Creating a Python Virtual Environment

```
$ python3 -m venv some_name
```

Listing 9. Activating the Environment

```
$ source some_name/bin/activate
```

Once you are done, you can deactivate your virtual environment with the command in Listing 10.

Listing 10. Deactivating Virtual Environment

```
$ deactivate
```

Creating a "requirements.txt" File

The "**requirements.txt**" file is used by most cloud providers to list any Python libraries needed for a hosted web application. In most cases, it is packaged alongside the web files and sent to its "**serverless**" destination for setup.

You can create your own "**requirements.txt**" and house it in the same folder as you main Flask Python script. Let's see how we can use virtual environments to create a complete "**requirements.txt**" file. When using a virtual environment, you are creating a safe sandbox free of any Python libraries. This allows you to install only what you need and run a "**pip freeze**" command to get a snapshot of the libraries and current version numbers. Mind you, you don't need to do this if you already know what libraries, dependencies, and version numbers you need. As a matter of fact, you can use the "**requirements.txt**" files packaged with this book content just as well.

Step 1

Start with a clean slate by creating a virtual environment in Python as shown in Listing 11.

Listing 11. Starting Virtual Environment

```
$ python3 -m venv some_env_name
$ source some_env_name/bin/activate
```

Step 2

Use "**pip3**" to install libraries needed to run a local web application, as shown in listing 12.

Listing 12. Installing Some Libraries as an Example

```
$ pip3 install flask
$ pip3 install pandas
$ pip3 install sklearn
```

Step 3

Freeze the environment and pipe all installed Python libraries, including version numbers in the "**requirements.txt**" file, as shown in Listing 13.

Listing 13. Installed Required Libraries

```
$ pip3 freeze > requirements.txt
```

Step 4

Finally, deactivate your virtual environment, as shown in Listing 14.

Listing 14. Deactivate out of venv

```
$ deactivate
```

There you go: you've just created a "**requirements.txt**" file. Check out its content by calling "**vi**" (click Escape and q to exit). The contents of your "**requirements.txt**" may look very different, and that's OK (Listing 15).

Listing 15. Checking the Content of "**requirements.txt**" File

Input:

```
$ vi requirements.txt
```

Ouput:

```
click==6.7
Flask==0.12.2
itsdangerous==0.24
Jinja2==2.10
MarkupSafe==1.0
numpy==1.14.2
scikit-learn
scipy
python-dateutil==2.7.2
pytz==2018.4
six==1.11.0
```

```
Werkzeug==0.14.1
Pillow>=1.0
matplotlib
gunicorn>=19.7.1
wtforms>=2.1
```

Inside the requirements.txt file, you can require a specific version by using the "==" sign (Listing 16)

Listing 16. Exact Assignment

```
Flask==0.12.2
```

You can also require a version equal to and larger, or equal to and smaller (Listing 17)

Listing 17. Directional Assignment

```
Flask >= 0.12
```

Or you can simply state the latest version that the installer can find (Listing 18)

Listing 18. Use Latest Version Available

```
Flask
```

Conclusion

This was only meant to be a brief introduction to the tools used in this book. Use these as jumping off points to explore further and to deepen your knowledge in the areas that are of particular interest to you.